UNITED STATES PATENT APPLICATION
FOR

METHOD AND APPARATUS FOR
REDIRECTING THE BOOT OPERATIONS OF ONE OR MORE SYSTEMS

INVENTORS:

MOTI HAIMOVSKY
EMEK-AYALON 60 ST., MODI'IN 71700, ISRAEL, A CITIZEN OF ISRAEL

YEHEZKEL LEVI
16 BIALIK ST., BAT-YAM 59471, ISRAEL, A CITIZEN OF ISRAEL

DORON OZ
18 VATIKIM STR., EVEN-YEHUDA, ISRAEL, A CITIZEN OF ISRAEL

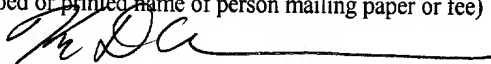### EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number: <u>EL759023031US</u>
Date of Deposit: <u>January 8, 2002</u>
I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to BOX PATENT APPLICATION, ASST COMMISSIONER FOR PATENTS, WASHINGTON DC 20231.

Kirk D. Williams
(Typed or printed name of person mailing paper or fee)

(Signature of person mailing paper or fee)

Jan 8, 2002
(Date signed)

METHOD AND APPARATUS FOR REDIRECTING THE BOOT OPERATIONS OF ONE OR MORE SYSTEMS

5

FIELD OF THE INVENTION

This invention especially relates to system initialization procedures of communications and computer systems; and more particularly, the invention relates to redirecting the boot operations of one or more systems.

10

BACKGROUND OF THE INVENTION

Embedded systems are usually computer-based systems, and are designed and

15   implemented for performing a predefined and specific activity. As such, their designs and complexities vary greatly, although they do share some common hardware and software structures. An embedded system, as well as a typical computer system, has several major hardware components connected together by buses. These components may include central processing units (CPUs), peripheral devices providing the interface between the

20   system an the external world, buses (e.g., CPU bus, PCI bus, SDRAM bus, etc.) connecting various system components among themselves and sometimes between systems that reside close to each other, and system controllers implementing various bus interfaces in the system, providing interrupt and timer support, and providing address decoding and re-mapping.

25   The basic operations performed in a computer system are fetch (or read) from a specified location and store (or write) to a specified location. Each location is identified by a unique address assigned to it in the bus it is located in, hence creating an address map for that bus, and when combined with other subsystems, an address map is defined for the whole system. However, each bus may view the system differently. For example,

devices may reside on different buses and have different address schemes and electrical characteristics. Yet, these diverse devices may need to communicate among themselves.

When a device on bus A (for example a PCI bus) wishes to access a device in bus B (for example a device bus), it places on its bus a request including the address of the

5   device it wishes to access. It is the responsibility of the system controller to identify that request, identify the bus for which the request is intended (also called address decoding), and to perform the necessary electrical, address and data conversions for placing the request on the destination bus.

For example, a CPU, in order to access a device residing in the PCI bus, puts its

10   request on the CPU bus. The system controller identifies that request, decodes the address, and determines that it is an address assigned to the PCI bus. The system controller further performs the necessary conversions and places the converted request on the PCI bus to be handled by the destination PCI device.

In terms of initializing such a system, after a processor has been reset, it has no

15   knowledge of the system it is in, therefore it always accesses to a specific address (0xFFF00100 in power PC) and executes the command written there. It is the responsibility of the designer to place the boot code in that address. And, more precisely, to preset the system controller such that this address when placed on the CPU bus will be transferred to the bus where the boot code resides.

20   In a PC, this is the task of the BIOS as can be seen on the screen when booting the PC. System boot process includes among other things, programming the system controller for correct address space decoding, performing power-on self-test, initializing various components in the system, obtaining the system image, creating the sufficient environment for booting it, and then transfer control to the operating system.

25   One of the issues in a multi-processor embedded environment, such as in a communications or computer device, is the software boot sequence. This issue is especially noticed in master-slaves systems where one central processor is used to manage the system and to synchronize the software versions and software download

processes of its slaves. In such systems, it is common to find a small and robust boot-loader software which is embedded into each processor module and is being executed by the local processor at boot time. The major task of such a boot-loader software is usually to perform the preliminary initializations (both software and hardware) required for

5   creating a communication channel with the master, synchronize themselves with the master software, and then wait for further instructions from the master which usually starts loading the "real" software to the slaves, a cumbersome process by itself. Although a distributed boot-loader is implemented in many systems, it is far from being trivial, and requires careful planning of software upgrade process, error recovery, inter boot-loaders

10   communication and more. Needed are new methods and apparatus for booting processors in multiprocessor and other systems.

SUMMARY OF THE INVENTION

Systems and methods are disclosed for redirecting the boot operations of one or more systems. One embodiment includes a master system coupled to a slave system. The slave system includes a programmable interface. A storage mechanism external to the

5   slave system includes a remote boot image. The master system updates the programmable interface of the slave system to retrieve the remote boot image. The slave system boots from the remote boot image. In one embodiment, the programmable interface of the slave controller includes a system controller, which is programmed by the master system to redirect a boot image retrieval request to the storage mechanism including the remote

10   boot image. In one embodiment, a corresponding update is made to the system controller of the master system. In one embodiment, multiple boot requests of multiple slave systems are redirected to the same or different boot images.

15

BRIEF DESCRIPTION OF THE DRAWINGS

The appended claims set forth the features of the invention with particularity. The invention, together with its advantages, may be best understood from the following detailed description taken in conjunction with the accompanying drawings of which:

5   FIG. 1 is a block diagram of an embodiment redirecting the boot operations of one or more systems;

FIG. 2 is a flow diagram of a process used in one embodiment for redirecting the boot operations of one or more systems; and

FIG. 3 is a message sequence chart illustrating the flow of a redirected boot

10   request and corresponding remote boot image in one embodiment.

DETAILED DESCRIPTION

Methods and apparatus are disclosed for redirecting the boot operations of one or more systems. Embodiments described herein include various elements and limitations, with no one element or limitation contemplated as being a critical element or limitation.

5 Each of the claims individually recite an aspect of the invention in its entirety. Moreover, some embodiments described may include, but are not limited to, *inter alia*, systems, networks, integrated circuit chips, embedded processors, ASICs, methods, and computer-readable medium containing instructions. The embodiments described hereinafter embody various aspects and configurations within the scope and spirit of the

10 invention, with the figures illustrating exemplary and non-limiting configurations.

As used herein, the term "system" is used generically herein to describe any number of components, elements, subsystems, processors, devices, packet switch elements, packet switches, routers, networks, computer and/or communication devices or mechanisms, or combinations of components thereof. The term "computer" is used

15 generically herein to describe any number of computers, including, but not limited to personal computers, embedded processors and systems, control logic, ASICs, chips, workstations, mainframes, etc. The term "device" is used generically herein to describe any type of mechanism, including a computer or system or component thereof. The terms "task" and "process" are used generically herein to describe any type of running program,

20 including, but not limited to a computer process, task, thread, executing application, operating system, user process, device driver, native code, machine or other language, etc., and can be interactive and/or non-interactive, executing locally and/or remotely, executing in foreground and/or background, executing in the user and/or operating system address spaces, a routine of a library and/or standalone application, and is not limited to

25 any particular memory partitioning technique. The steps, connections, and processing of signals and information illustrated in the figures, including, but not limited to the block and flow diagrams, are typically performed in a different serial or parallel ordering and/or by different components and/or over different connections in various embodiments in

keeping within the scope and spirit of the invention. Moreover, the terms "network" and "communications mechanism" are used generically herein to describe one or more networks, communications mediums or communications systems, including, but not limited to the Internet, private or public telephone, cellular, wireless, satellite, cable, local

5   area, metropolitan area and/or wide area networks, a cable, electrical connection, bus, etc., and internal communications mechanisms such as message passing, interprocess communications, shared memory, etc. The term "storage mechanism" includes any type of memory, storage device or other mechanism for maintaining instructions or data in any format. The term "data structure" is an extensible term referring to any data structure, data

10   base, and/or one or more or an organizational schemes that can be applied to data to facilitate interpreting the data or performing operations on it, such as, but not limited to memory locations or devices, sets, trees, heaps, lists, linked lists, arrays, tables, pointers, etc. The terms "first," "second," etc. are typically used herein to denote different units (e.g., a first element, a second element). The use of these terms herein does not

15   necessarily connote an ordering such as one unit or event occurring or coming before the another, but rather provides a mechanism to distinguish between particular units. Moreover, the phrase "based on x" is used to indicate a minimum set of items x from which something is derived, wherein "x" is extensible and does not necessarily describe a complete list of items on which the operation is based. Additionally, the phrase "coupled

20   to" is used to indicate some level of direct or indirect connection between two elements or devices, with the coupling device or devices modify or not modifying the coupled signal or communicated information. The term "subset" is used to indicate a group of all, less than all, or none of the elements of a set. Moreover, the term "or" is used herein to identify an alternative selection of one or more, including all, of the conjunctive items.

25   Methods and apparatus are disclosed for redirecting the boot operations of one or more systems. In one embodiment, a master boot processor configures one or more slave boot processors to boot over a communication bus existing between the processors, which typically eliminates the need for boot software in the slave systems.

One embodiment includes a master system coupled to a slave system. The slave system includes a programmable interface. A storage mechanism external to the slave system includes a remote boot image. The master system updates the programmable interface of the slave system to retrieve the remote boot image. The slave system boots

5   from the remote boot image. In one embodiment, the programmable interface of the slave controller includes a system controller, which is programmed by the master system to redirect a boot image retrieval request to the storage mechanism including the remote boot image. In one embodiment, a corresponding update is made to the system controller of the master system. In one embodiment, multiple boot requests of multiple slave

10   systems are redirected to the same or different boot images.

In one embodiment, one or more configurable components are included in the master and/or slave systems, such that, these can be configured by the master without software assistance from the slave. These components are capable of transforming and redirecting one bus protocol and addresses to another one. In this way, every slave-CPU

15   access, starting from the first command it performs, is automatically being translated and transformed to and through the communication bus to the slave software residing in the master system (e.g., in memory or other storage device) allowing the slave to directly boot from software located there. Because this is a configurable component, it also very flexible in address translation between buses. Each slave can be directed to boot from a

20   same or different location. One embodiment provides the master a simple way for controlling the software being loaded by each slave by redirecting the boot operations of one or more systems. In one embodiment, these slave programmable interfaces are statically preprogrammed. In one embodiment, these slave programmable interfaces are dynamically programmed by a master or other system or device.

25   In one embodiment, slave systems use the same boot software, thus, it can be upgraded by replacing a single image of the software in a memory or storage device of the slave boot source. In one embodiment, the slave boot source resides within the master system. In one embodiment, the slave boot source resides in another of the slave systems,

8

or in another location or system (e.g., memory or storage device in another system or on a bus.) Thus, the booting of one or more slaves (and possibly the master) is redirected to the other system wherein the boot software resides. In one embodiment, at least one of the slave systems is redirected to boot from a first location different than a second location

5   where another of the slave systems is redirected from which to boot. In one embodiment the first and second locations are different addresses within a same memory or storage device. In one embodiment, one or more slave systems are booted sequentially. In one embodiment, one or more slave systems are booted from the same boot source approximately simultaneously. In one embodiment, the number or size of non-volatile

10   memory required by a slave system is reduced as it is no longer required for booting.

In one embodiment, a device includes six stand-alone embedded systems, with one of the controllers acting as the master and the other as slaves. The boot software for all slaves resides in memory or a storage device included in the master system, and the slaves are configured to boot from it while thinking that they are booting from a local

15   boot device. In one embodiment, all the slave devices are connected to the master via a PCI bus using a programmable Galileo GT64240 system controller. Using the GT64260 mapping registers, the master programs the slaves to direct the CPU 0xFF000000 address and above to the PCI, and programs its controller to direct those addresses to the bus where the boot code (also herein referred to as the boot image) resides, hence creating a

20   tunnel between each of the slaves CPUs and the device where the boot code resides. Releasing a slave CPU after this configuration has been performed, the slave CPU will boot as directed with the boot code residing in the master, other device or location. Using such a configurable component, allows the boot process to be very flexible in address translation between buses. Dynamically, a slave can be directed to boot from a different

25   location, which provides the master a simple way for controlling the software being loaded by each slave.

FIG. 1 illustrates one embodiment of an apparatus with a master system 100 redirecting the boot operations of one or more slave systems 110 and 120, and/or other

slaves or other devices 141, 142, and 143. Systems 100, 110 and 120 may be part of a router or other communications or computer system, or any other device. In one embodiment, these systems and devices are interconnected via one or more communications mechanisms 109A, 109B and 119 (shown as buses for illustrative

5   purposes) and systems controllers (e.g., system controllers 104, 114 and/or 124). Although two slave systems 110 and 120 are illustrated in detail, it is to be understood that one embodiment includes only one slave system, while one embodiment includes more than two slave systems.

In one embodiment, master system 100 includes a processor 101, memory 102,

10   storage devices 103, and a system controller 104. Various embodiments of system 100 may include more or less elements. For example, one embodiment does not include storage devices 103. The operation of system 100 is typically controlled by processor 101 using memory 102 and storage devices 103 to perform one or more tasks or processes. Memory 102 is one type of computer-readable medium, and typically comprises random

15   access memory (RAM), read only memory (ROM), flash memory, integrated circuits, and/or other memory components. Memory 102 typically stores computer-executable instructions to be executed by processor 101 and/or data which is manipulated by processor 101 for implementing functionality in accordance with embodiments of the invention. In one embodiment, memory 102 store one or more remote boot images which

20   will may be used for booting slave system 110 and/or 120. Storage devices 103 are another type of computer-readable medium, and typically comprise solid state storage media, disk drives, diskettes, networked services, tape drives, and other storage devices. Storage devices 103 typically store computer-executable instructions to be executed by processor 101 and/or data which is manipulated by processor 101 for implementing

25   functionality in accordance with embodiments of the invention. In one embodiment, storage devices 103 store one or more remote boot images which will may be used for booting slave system 110 and/or 120.

As used herein and contemplated by the invention, computer-readable medium is not limited to memory and storage devices; rather computer-readable medium is an extensible term including other storage and signaling mechanisms including interfaces and devices such as network interface cards and buffers therein, as well as any

5   communications devices and signals received and transmitted, and other current and evolving technologies that a computerized system can interpret, receive, and/or transmit.

FIG. 1 further illustrates one embodiment of slave system 110. In one embodiment, slave system 110 includes a processor 111, memory 112, storage devices 113, and a system controller 114. Various embodiments of slave system 110 may

10   include more or less elements. For example, one embodiment does not include storage devices 113. The operation of system 111 is typically controlled by processor 111 using memory 112 and storage devices 113 to perform one or more tasks or processes. In one embodiment, memory 112 and/or storage devices 113 store one or more remote boot images which will may be used for booting slave system 120.

15   FIG. 1 further illustrates one embodiment of slave system 120. In one embodiment, slave system 120 includes a processor 121, memory 122, storage devices 123, and a system controller 124. Various embodiments of slave system 120 may include more or less elements. For example, one embodiment does not include storage devices 123. The operation of system 121 is typically controlled by processor 121 using

20   memory 122 and storage devices 123 to perform one or more tasks or processes. In one embodiment, memory 122 and/or storage devices 123 store one or more remote boot images which will may be used for booting slave system 110.

In one embodiment, master system 100 is first booted, with slave processors 111 and 121 remaining in a reset condition until released by master processor 101. After

25   master system 110 has completed its boot process (or at least got into a well defined state), it initiates the booting of one or more of the slave systems 110 and 120 (and any other slaves, devices, etc. in the same manner) after manipulating slave systems 110 and 120 to basically believe that their boot code is located locally in their respective system,

while it actually resides in another location, such as in master (volatile or non-volatile) memory 102 or storage devices 103, memory or storage devices 131 (accessed via controller 134), or memory 112, 122 or storage devices 113, 123 of another of the slave systems 110, 120, or other locations.

5   FIG. 2 presents a flow diagram of a boot process used in one embodiment for redirecting the boot operations of one or more systems. At power-up or upon reset or other initialization condition, processing begins with process block 200, and proceeds to process block 202, wherein the master system is booted. In process block 204, the slave controllers are released, and as indicated in process block 206, a possible delay is

10   imposed to ensure their initialization has stabilized.

Next, in process block 208, the slave systems whose boot operation is to be redirected are identified. In one embodiment, the shared bus or buses (e.g., PCI buses) are scanned or predefined addresses are used in order to locate the slave devices. In one embodiment, the PCI bus is scanned looking for the system controllers of the slaves. For

15   each slave found, the master interrogates the slave for its type and version, while in one embodiment, this information is predetermined and available to the master system such as via a data structure located in memory. In process block 210, the master system determines the boot software and its location appropriate for the slaves and the particular application to be performed by each slave. This software can reside in a local or remote

20   memory, storage device, server, or other location.

In one embodiment, as indicated by process blocks 212 and 214, if the corresponding one or more boot software sets are not located in memory, the corresponding one or more boot software sets are retrieved and stored in memory for easy and quick access during the booting of the slave systems.

25   Next, in process block 216, the master system assigns each slave controller with an address or a range of addresses to be used by the master processor for accessing the slave systems. In process block 218, the master programs its own system controller with these addresses. From this point on, when the master processor issues an address destined

to a certain slave, the master system controller will translate it to an appropriate bus address and redirect it to the bus where the desired slave controller is located.

In process block 220, the master processor programs each of the slave system controllers to allow access to their internal registers from the shared bus to allow the

5   master processor to configure the slave system controllers. In one embodiment, the master processor, using PCI configuration cycles, accesses the PCI registers of the slave system controllers and assigns each slave system controller with the base address and an access rule. From this point on, the master processor can access the registers of the slave system controllers by issuing standard read/write commands from/to those addresses.

10   In process block 222, the master processor manipulates the registers of the slave system controllers so that when a slave processor issues an address that is part of the boot-code range of addresses, this address will be directed the shared bus. In one embodiment, the system controllers are also programmed to translate the address to a different one when one or more slaves are to access one or more different locations in the

15   master system. In this manner, the addresses generated by the slave CPU, which were originally directed to its boot ROM, are now redirected to the shared bus and to the desired remotely located boot code.

In process block 224, the master programs its system controller to direct the slave boot addresses (generated by a slave CPU and directed by a corresponding slave system

20   controller to the shared bus) to the corresponding boot code location, which may reside in memory or storage device of the master system, another slave system, remote server, or other system or device.

In process block 226, the processors of the slave systems are released, so they will start their boot process, and retrieve its boot code from the redirected location. This will

25   eventually boot the slaves. Processing is complete as indicated by process block 228.

FIG. 3 illustrates a message sequence chart illustrating the flow of a redirected boot request and corresponding remote boot image in one embodiment. Message sequence charts are well-known and often used to illustrate communication of any type

and format of information (e.g., messages, data, instructions, files, etc.) between components or systems. In one embodiment, slave system controller 302 and remote system controller 303 are programmed to redirect a boot image retrieval request for the slave system to remote storage mechanism 304, which contains the remote boot image.

5   In one embodiment, slave processor 301 initiates a boot image retrieval request 311, which is identified by slave system controller 302, which then relays boot image retrieval request 312 to remote system controller 303, which then relays boot image retrieval request 313 to remote storage mechanism 304. Remote storage mechanism 304 retrieves the remote boot image in response, and sends the remote boot image 314 to

10   remote system controller 303, which then relays remote boot image 315 to slave system controller 302, which then delivers remote boot image 316 to slave processor 301 (or a memory associated therewith). Slave processor 301 then boots using remote boot image 316.

In view of the many possible embodiments to which the principles of our

15   invention may be applied, it will be appreciated that the embodiments and aspects thereof described herein with respect to the drawings/figures are only illustrative and should not be taken as limiting the scope of the invention. For example and as would be apparent to one skilled in the art, many of the process block operations can be re-ordered to be performed before, after, or substantially concurrent with other operations. Also, many

20   different forms of data structures could be used in various embodiments. The invention as described herein contemplates all such embodiments as may come within the scope of the following claims and equivalents thereof.